LA-UR-19-32580

Title:	A simple json format for storing and exchanging code calculation and experimental results
Author(s):	Haeck, Wim
Intended for:	Report
Issued:	2019-12-18

A simple json format for storing and exchanging code calculation and experimental results

W. Haeck

Los Alamos National Laboratory, Los Alamos, NM, 87545, USA

December 11, 2019

Contents

1 Introduction **2**

2 JSON structure **4**

2.1 JSON Schema 4

2.2 Result 4

2.3 Attributes 5

2.4 Data 6

2.5 Dimension 8

2.6 Units 8

2.7 Examples 9

3 Python interface **14**

References **16**

Appendices **17**

A JSON schema **18**

Chapter 1

Introduction

Calculation codes and code systems give their results in a plethora of different formats and/or files. For instance, MCNP [1] gives a user the MCNP specific output file, the mctal file, the ptrac file, etc. while PARTISN [2] has a PARTISN specific output file and a number of binary files with various kinds of information about the calculation.

However, the results we are interested in are always the same. These can be single values such as the effective multiplication factor k_{eff} or the effective delayed neutron fraction β_{eff}. It can be histogram data (such as particle spectra, reaction rates, sensitivity profiles, etc.) or even pointwise data (for example the nuclide composition as a function of time, etc.). It is also worthwhile to note that this would also apply to experimental data that we may wish to compare with as well.

Calculation and experimental results can basically be split into two components: attributes (or metadata) and the actual result itself. Attributes represent information about the result that are not necessarily required for its' interpretation but that are still good to have. Often, these attributes are what we will want to filter results on. This means that these attributes are often the answer to specific questions we may have about the result. The following is a non exhaustive list of such basic questions:

- What type of result is it: e.g. the effective multiplication factor k_{eff}, the effective delayed neutron fraction β_{eff}, etc.
- Which case does it relate to: e.g. a specific ICSBEP case such as PU-MET-FAST-001-001
- Which code and version produced this result: e.g. MCNP 6.2, etc.
- Which nuclear data library was used to obtain the result: e.g. ENDF/B-VIII.0 or JEFF 3.3
- Which nuclide and reaction does it apply to (for reaction rates and sensitivity profiles)?
- Which volume in the geometry was it calculated in?

The second component of the calculation or experimental result is the actual numerical data of the result. It is this data that we want to compare, exchange, plot, interpret, etc. It consists of values with optional uncertainties and units. It also includes the underlying structure (if any structure is required to interpret the result). Depending on the type of result, it may be a single value (e.g. the effective multiplication factor), an array of values (e.g. a reaction rate or particle spectrum) or even a multidimensional array.

In order to exchange results between different organisations, for instance for benchmarking, people often resort to spreadsheets or custom text files that need to be filled out. While this approach works well if little information is to be exchanged, it can become cumbersome when dealing with large amounts of data (like for instance particle spectra for thousands of cases). In addition, such an approach does not always capture the metadata we may be interested in to allow for automated processing of these files.

The purpose of this report is to propose a standardised json structure to store any type of result and present an associated python interface to interact with these results directly. For such a structure to be useful, it must fulfill a specific set of functional requirements that we try to meet with this proposal:

- The format must allow for the calculation result to stand by itself. This means that we should not have to look in multiple places to be able to understand it. For example, if the result relates to a particle spectrum, the group structure in which the result is given should be stored with it.
- The format should be code agnostic. This means that we do not need to know where it came from to understand it.
- The format should be result type agnostic. This means that the format must be generic enough so that we can even use it for results we have not thought of yet.
- The format should be relatively easy to interact with through different scripting or programming languages (i.e. the machine interface) while still providing a clean structure that can be used directly by any human.

Chapter 2

JSON structure

2.1 JSON Schema

JSON Schema is a vocabulary that allows us to annotate and validate JSON documents. While it is not yet recognised a full standard, a large number of tools already support it (e.g. python has a jsonschema package that can use JSON Schema). One advantage is that it provides clear human- and machine-readable documentation, which makes it the preferred way of documenting a JSON structure, as this schema can then be used in automated testing and ensure quality (and compliance) of client submitted data.

The following websites are good resources for JSON Schema:

- JSON Schema documentation
- JSON object validator using JSON Schema

A JSON schema was produced for the proposed JSON structure for calculation results and can be found in Appendix A. The examples given in section 2.7 have been tested using the webtool cited above.

2.2 Result

JSON object structure:

```
{
  "attributes" : { ... },
  "data" : { ... }
}
```

Property	Type	Description	Required	Page
`"attributes"`	object	Attributes (or metadata) of the result	yes	5
`"data"`	object	Data for the result	yes	6

The top level JSON object representing a calculation or experimental result only has two properties (`"attributes"` and `"data"`), both of which are required. `"attributes"` contains the attributes (or metadata) about the result while `"data"` contains the actual numerical result (potentially with uncertainties, units and dimensions).

2.3 Attributes

JSON object structure:

```
{
  "type" : "...",
  "case" : "...",
  "code" : "...",
  "library" : "...",
  "particle" : "...",
  "nuclide" : "...",
  "reaction" : "...",
  "material" : "...",
  "response" : "...",
  "parameter" : "..."
}
```

Property	Type	Description	Required	Page
"type"	string	The type of the result (a number of possible values have already been identified, see below)	yes	-
"case"	string	The case or calculation identifier (e.g. an ICSBEP name such as 'HEU-MET-FAST-001-001-s' or another user defined name) for which this result was generated	yes	-
"code"	string	The calculation code name and version that generated the result (e.g. 'MCNP6.2')	no	-
"library"	string	The nuclear data library used to generate the result (e.g. 'ENDF/B-VIII.0')	no	-
"particle"	string	The incident particle associated to the result (e.g. 'neutron')	no	-
"nuclide"	string	The nuclide to which the result applies (e.g. 'U235')	no	-
"reaction"	string	The reaction to which the result applies (e.g. 'elastic')	no	-
"material"	string	The material to which the result applies (if this is the entire geometry, this is set to 'total')	no	-
"response"	string	The response type to which the sensitivity $S = \frac{\partial r}{\partial p}$ applies (these should refer to other result types so that we can link to those results)	no	-
"parameter"	string	The parameter to which the sensitivity profile applies $S = \frac{\partial r}{\partial p}$ applies (e.g. 'crossSection', 'angular' or 'spectrum')	no	-

The attributes of the result represent information or metadata associated to the result. It is this type of information we may want to filter on when we have a list of these results. For example, we may be interested in filtering a list of results to obtain all sensitivity profiles to U235 cross sections for a given benchmark case. Essentially, any key,value pair is allowed in here but there are already a few predefined attributes. The list of predefined attributes given here is in no way complete and additional ones will be added once the need for them arises.

In the case of "type", a number of possible values for "type" have already been identified, along with which attributes may be required when such a result type is given. This is detailed in Table 2.1.

2.4 Data

JSON object structure:

```
{
  "values" : [ 1.0, ..., 5.0 ],
  "uncertainties" : [ 1.0, ..., 5.0 ],
  "structure" : [ { ... }, ..., { ... } ],
  "units" :  ...
}
```

Table 2.1: Overview of attribute "type" values and associated additional attributes.

Type	Additional attributes
"effectiveMultiplicationFactor"	-
"effectiveDelayedNeutronFraction"	-
"effectiveNeutronGenerationTime"	-
"promptNeutronDecayConstant"	-
"thermalFissionFraction"	-
"aboveThermalFissionFraction"	-
"fissionFractions"	-
"energyAverageLethargyFission"	-
"averageEnergyCausingFission"	-
"particleSpectrum"	"particle", "material"
"reactionRate"	"particle", "nuclide", "reaction", "material"
"timeOfFlightSpectrum	"particle"
"sensitivityProfile"	"particle", "nuclide", "reaction", "material", "response", "parameter"

Property	Type	Description	Required	Page
"values"	array	Numerical values that represent the result, given in a flat array (if there is a structure to these values, that structure is given as a separate attribute)	yes	-
"uncertainties"	array	Uncertainties associated to the numerical values (if this is used, this must be of the same size as the values array)	no	-
"structure"	array	An array of dimension objects that describes the structure of the values and uncertainties (if this is used, one or more dimension objects must be given)	no	8
"units"	object	The units in which the values and uncertainties are given	no	8

The result data represents the numerical result and associated information such as uncertainties, units, group structures (if such information is required to fully understand the result). Only the "values" attribute is required and it is always an array of values, even if the result is a single numerical value (as would be the case for an effective multiplication value or any similar result). If the "values" array contains more than one value, the use of the "structure" array of dimensions becomes required.

The other properties are optional but there are some requirements in their use. If the "uncertainties" property gives an array of associated uncertainties, both the "values" and "uncertainties" arrays must have the same size. In addition, if an array of dimensions is given, then the number of values derived from these dimensions must be the size of the "values" and "uncertainties" arrays (see section 2.6 for more information).

As mentioned above, the values are given as a single flat array so dimensions are used to define the underlying structure. A one-dimensional result (e.g. a particle spectrum integrated over a given number of incident energy bins) will have only one dimension. The order in which these dimensions are given determines the order in which the values are interpreted. When going through dimensions, each bin or point of the current dimension contains the data for the following dimension. For example, if an incident energy dimension is given with 5 bins followed by an outgoing energy dimension with 2 bins, then every sequential pair in the "values" array are the corresponding values for the outgoing energy dimension for each of the incident energy bins.

Dimensions can also be mixed. For instance, one could imagine storing a set of points in time and give a histogram (e.g. a particle spectrum) for each point.

2.5 Dimension

JSON object structure:

```
{
  "name" : "...",
  "type" : "...",
  "limits" : [ 1.0, ..., 5.0 ],
  "units" : { ... }
}
```

Property	Type	Description	Required	Page
"name"	string	The name of the dimension (e.g. 'energy-in')	yes	-
"type"	string	The type of the dimension (either 'histogram' or 'points')	yes	-
"limits"	array	The histogram bins or points for which data will be given in this dimension (these are generally numbers but they could be strings, e.g. when labels are used)	yes	-
"unit"	object	The unit associated to this dimension (e.g. 'MeV')	no	8

Dimensions are used to defined the structure of the data given in the results. The "name", "type" and "limits" property are required when defining a dimension. The value for "name" can be set arbitrarily (it was added for plotting purposes). There are two possibilities for the "type" property: either histogram or points. This value for "type" will determine how many values have to be present (the number of values in the "limits" array minus one for the "histogram" type and the number of values when the type is "points").

2.6 Units

JSON object structure:

```
{
  "value" : "...",
  "uncertainty" : "..."
}
```

Property	Type	Description	Required	Page
"value"	string	The unit of the values	yes	-
"uncertainty"	string	The unit of the uncertainties	no	-

These units are optional and are used to give the unit of the the values (and uncertainties if they are given). These units are given as strings. Currently, we have no predefined values for these units but they may be added should the need arise.

2.7 Examples

The following are examples for results for a calculation on HEU-MET-FAST-001-001-s from an MCNP6.2 calculation using ENDF/B-VIII.0.

The following is an example of a result storing only a single effective multiplication factor value and its uncertainty. It represents the simplest instance of this JSON result structure.

```
{
  "attributes": {
    "type": "effectiveMultiplicationFactor",
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2",
    "library": "ENDF/B-VIII.0"
  },
  "data": {
    "values": [ 0.99993 ],
    "uncertainties": [ 9e-05 ]
  }
}
```

The following are results for the average energy causing fission and the energy corresponding to the average neutron lethargy causing fission. Compared to the previous example, these results add units for the value given in the results.

```
{
  "attributes": {
    "type": "averageEnergyCausingFission",
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2",
    "library": "ENDF/B-VIII.0"
  },
  "data": {
    "values": [ 1.4571 ],
    "units": {
      "value": "MeV"
    }
  }
}
```

```
{
  "attributes": {
    "type": "energyAverageLethargyFission",
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2",
    "library": "ENDF/B-VIII.0"
  },
  "data": {
    "values": [ 0.82893 ],
    "units": {
      "value": "MeV"
    }
  }
}
```

The following are fission fraction results. They could be stored as single values (which is the case for the thermal fission fraction and the above thermal fission fraction) or as a list of values with an associated group structure.

```
{
  "attributes": {
    "type": "thermalFissionFraction",
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2",
    "library": "ENDF/B-VIII.0"
  },
  "data": {
    "values": [ 0.0 ],
    "units": {
      "value": "%"
    }
  }
}
```

```
{
  "attributes": {
    "type": "aboveThermalFissionFraction",
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2",
    "library": "ENDF/B-VIII.0"
  },
  "data": {
    "values": [ 100.0 ],
    "units": {
      "value": "%"
    }
  }
}
```

```
{
  "attributes": {
    "type": "fissionFractions",
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2",
    "library": "ENDF/B-VIII.0"
  },
  "data": {
    "values": [ 0.0, 4.81, 95.19 ],
    "structure": [
      {
        "name": "energy",
        "type": "histogram",
        "limits": [ 1e-11, 6.25e-07, 100000.0, 20000000.0 ],
        "unit": "MeV"
      }
    ],
    "units": {
      "value": "%"
    }
  }
}
```

In addition to the effective multiplication factor, MCNP6.2 can also provide the point kinetics parameters. The structure of these results are similar to ones already given above. Th effective delayed neutron fraction is essentially similar to the effective multiplication factor while the layout of the result storing the effective neutron generation time and the Rossi-alpha value (the prompt neutron decay constant) is similar to the single value fission fraction results.

```
{
  "attributes": {
    "type": "effectiveDelayedNeutronFraction",
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2",
    "library": "ENDF/B-VIII.0"
  },
  "data": {
    "values": [ 0.00652 ],
    "uncertainties": [ 7e-05 ]
  }
}
```

```
{
  "attributes": {
    "type": "effectiveNeutronGenerationTime",
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2",
    "library": "ENDF/B-VIII.0"
  },
  "data": {
    "values": [ 5.62752 ],
    "uncertainties": [ 0.00713 ],
    "units": {
      "value": "ns",
      "uncertainty": "ns"
    }
  }
}
```

```
{
  "attributes": {
    "type": "promptNeutronDecayConstant",
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2",
    "library": "ENDF/B-VIII.0"
  },
  "data": {
    "values": [ -0.00115865 ],
    "uncertainties": [ 1.18917e-05 ],
    "units": {
      "value": "1/ns",
      "uncertainty": "1/ns"
    }
  }
}
```

The following is an example of a sensitivity profiles for U235 delayed ν. JSON is very flexible, we do not care about the order as long as the required properties are present. This is illustrated in this example (the data precedes) the result attributes.

```
{
  "data": {
    "values": [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                0.0, 0.0, 1.6904e-08, 1.2892e-08,
                9.3397e-08, 1.2908e-06, 1.7417e-05,
                1.6257e-05, 0.00028105, 0.0014029,
                0.0014216, 0.00093195, 0.00059893,
                0.00048545, 9.7822e-05, 0.00032994,
                0.00052972, 0.00011731, 4.0909e-05,
                1.7765e-05 ],
    "uncertainties": [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                       0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                       0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                       0.0, 0.0, 0.995, 0.8954, 0.2719, 0.1299,
                       0.0282, 0.0364, 0.0074, 0.0031, 0.0033,
                       0.0048, 0.0057, 0.0058, 0.0139, 0.0071,
                       0.0058, 0.0093, 0.0131, 0.0192 ],
    "structure": [
      {
        "name": "energy-in",
        "type": "histogram",
        "limits": [ 1e-11, 3e-09, 7.5e-09, 1e-08, 2.53e-08,
                    3e-08, 4e-08, 5e-08, 7e-08, 1e-07, 1.5e-07,
                    2e-07, 2.25e-07, 2.5e-07, 2.75e-07, 3.25e-07,
                    3.5e-07, 3.75e-07, 4e-07, 6.25e-07, 1e-06,
                    1.77e-06, 3e-06, 4.75e-06, 6e-06, 8.1e-06,
                    1e-05, 3e-05, 0.0001, 0.00055, 0.003, 0.017,
                    0.025, 0.1, 0.4, 0.9, 1.4, 1.85, 2.354, 2.479,
                    3.0, 4.8, 6.434, 8.1873, 20.0 ],
        "unit": "MeV"
      }
    ],
    "units": {
      "value": "%/%",
      "uncertainty": "relative"
    }
  },
  "attributes": {
    "type": "sensitivityProfile",
    "response": "effectiveMultiplicationFactor",
    "parameter": "xs",
    "particle": "neutron",
    "nuclide": "U235",
    "reaction": "delayed nu",
    "material": "total",
    "library": "ENDF/B-VIII.0"
    "case": "HEU-MET-FAST-001-001-s",
    "code": "MCNP6.2"
  }
}
```

Chapter 3

Python interface

A python interface has been developed to serialise/deserialise these results to/from JSON files and to more easily interact with it. Different classes were developed so that the interface mirrors the structure described above.

Attributes and metadata of the result can be accessed as follows. If a user asks for an attribute that is not defined, `None` will be returned. In these examples, the `result` variable refers to a single result (for this example, we will assume it is the sensitivity profile given in section 2.7).

```
# retrieve attribute information with the "attributes" property
type = result.attributes.type           # "sensitivityProfile"
nuclide = result.attributes.nuclide     # "U235"
reaction = result.attributes.reaction   # "delayed nu"
date = result.attributes.date           # None, "date" attribute is not defined
```

The data of the result can be accessed through the `data` property on the result:

```
# retrieve the numerical data with the data property
values = result.data.values
uncertainties = result.data.uncertainties
dimensions = result.data.structure
units = result.data.units
```

or they can be accessed through shortcut properties provided on the result:

```
# retrieve the numerical data with the "data" property
values = result.values
uncertainties = result.uncertainties
dimensions = result.structure
units = result.units
```

The units and dimensions provide similar interfaces:

```
# retrieve dimension information
print( len( result.structure ) )              # only 1 dimension
```

```
# dimension information
name = result.structure[0].name                # "energy-in"
type = result.structure[0].type                # "histogram"
groups = result.structure[0].limits            # [ 1e-11, ..., 20.0 ]
unit = result.structure[0].unit                # "MeV"

# units information
valueUnit = result.units.value                 # "%/%"
uncertaintyUnit = result.units.uncertainty     # "relative"
```

JSON serilisation and deserialisation functions are provided for single results, lists of results and dictionaries of results:

```
toJSON( result, 'mcnp.results.json', indent = 2 )
resultFromJSON = fromJSON( 'mcnp.results.json' )

print( result = resultsFromJSON )   # should be True
```

References

[1] D. B. Pelowitz, A. J. Fallgren, G. E. McMath, “MCNP6 User’s Manual - Code Version 6.1.1beta”, LA-CP-14-00745 Rev. 0, Los Alamos National Laboratory, USA (2014)

[2] R. E. Alcouffe, R. S. Baker, J. A. Dahl, S. A. Turner, R. C. Ward, “PARTISN: A Time-Dependent, Parallel Neutral Particle Transport Code System”, LA-UR-08-07258 Rev. November 2008, Los Alamos National Laboratory, USA (2008)

Appendices

Appendix A

JSON schema

Please note that line breaks were added to the JSON schema given below to make some of the description property values fit inside a page. To use this schema in a practical context, these line breaks will need to be removed first as JSON does not allow for strings to be spread over multiple lines.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "title": "A result consisting of attributes and numerical data",
  "required": [
    "attributes",
    "data"
  ],
  "properties": {
    "attributes": {
      "type": "object",
      "title": "Result attributes",
      "description": "Attributes (or metadata) of the result",
      "required": [
        "type",
        "case"
      ],
      "properties": {
        "type": {
          "type": "string",
          "description": "The type of the result (a number of possible values
                          have already been identified, see below",
          "examples": [
            "effectiveMultiplicationFactor",
            "effectiveDelayedNeutronFraction",
            "effectiveNeutronGenerationTime",
            "promptNeutronDecayConstant",
            "thermalFissionFraction",
            "aboveThermalFissionFraction",
            "fissionFractions",
            "energyAverageLethargyFission",
            "averageEnergyCausingFission",
            "sensitivityProfile",
    "particleSpectrum",
    "reactionRate",
    "timeOfFlightSpectrum"
  ]
},
"case": {
  "type": "string",
  "description": "The case or calculation identifier (e.g. an ICSBEP
                  name  such as ’HEU-MET-FAST-001-001-s’ or another user
                  defined name) for which this result was generated"
},
"code": {
  "type": "string",
  "description": "The calculation code name and version that generated
                  the result (e.g. ’MCNP6.2’)"
},
"library": {
  "type": "string",
  "description": "The nuclear data library used to generate the result
                  (e.g. ’ENDF/B-VIII.0’)"
},
"particle": {
  "type": "string",
  "description": "The incident particle associated to the result (e.g.
                  ’neutron’)"
},
"nuclide": {
  "type": "string",
  "description": "The nuclide to which the result applies (e.g. ’U235’)"
},
"reaction": {
  "type": "string",
  "description": "The reaction to which the result applies (e.g.
                  ’elastic’)"
},
"material": {
  "type": "string",
  "description": "The material to which the result applies (if this is
                  the entire geometry, this is set to ’total’)"
},
"response": {
  "type": "string",
  "description": "The response type to which the sensitivity applies
                  (these should refer to other result types so that we
                  can link to those results)",
  "examples": [
    "effectiveMultiplicationFactor",
    "effectiveDelayedNeutronFraction",
    "effectiveNeutronGenerationTime",
    "promptNeutronDecayConstant",
    "thermalFissionFraction",
    "aboveThermalFissionFraction",
    "fissionFractions",
    "energyAverageLethargyFission",
    "averageEnergyCausingFission",
    "sensitivityProfile",
          "particleSpectrum",
          "reactionRate",
          "timeOfFlightSpectrum"
        ]
      },
      "parameter": {
        "type": "string",
        "description": "The parameter to which the sensitivity profile
                        applies applies (e.g. ’crossSection’, ’angular’ or
                        ’spectrum’)",
        "examples": [
          "crossSection",
          "angular",
          "spectrum"
        ]
      }
    }
  },
  "data": {
    "type": "object",
    "title": "Result data",
    "description": "Data for the result",
    "required": [
      "values"
    ],
    "properties": {
      "values": {
        "type": "array",
        "description": "Numerical values that represent the result, given
                        in a flat array (if there is a structure to these
                        values, that structure is given as a separate attribute)",
        "items": {
          "type": "number"
        }
      },
      "uncertainties": {
        "type": "array",
        "description": "Uncertainties associated to the numerical values (if
                        this is used, this must be of the same size as the
                        values array)",
        "items": {
          "type": "number"
        }
      },
      "structure": {
        "type": "array",
        "description": "An array of dimension objects that describes the
                        structure of the values and uncertainties (if this is
                        used, one or more dimension objects must be given)",
        "items": {
          "type": "object",
          "title": "Dimension",
          "description": "A dimension in the data structure",
          "required": [
            "name",
            "type",
              "limits"
            ],
            "properties": {
              "name": {
                "type": "string",
                "description": "The name of the dimension (e.g. 'energy-in')"
              },
              "type": {
                "type": "string",
                "description": "The type of the dimension (either 'histogram'
                                or 'points')"
              },
              "limits": {
                "type": "array",
                "description": "The histogram bins or points for which data
                                will be given in this dimension (these are generally
                                numbers but they could be strings, e.g. when labels
                                are used)"
              },
              "unit": {
                "type": "string",
                "description": "The unit associated to this dimension (e.g. 'MeV')"
              }
            }
          }
        },
        "units": {
          "type": "object",
          "description": "The units in which the values and uncertainties are given",
          "items": {
            "type": "object",
            "title": "Dimension",
            "description": "A dimension in the data structure",
            "required": [
              "value"
            ],
            "properties": {
              "value": {
                "type": "string",
                "description": "The unit of the values"
              },
              "uncertainty": {
                "type": "string",
                "description": "The unit of the uncertainties"
              }
            }
          }
        }
      }
    }
  }
}
```